TP CLIENTS LEGERS SOUS THINSTATION

François Ducrot - Jacquelin Charbonnel

Journées Mathrice - Mars 2009 - Angers

1 Introduction

Thinstation est une petite distribution Linux bootant par PXE, et qui transforme un PC en client léger (CL). Ce CL peut ensuite être utilisé en autonome, ou être utilisé pour se connecter à un serveur applicatif via divers protocoles. La solution Thinstation met en oeuvre :

- des PC destinés à être des CL,
- un serveur d'infrastructure nécessaire au boot des CL, c'est-à-dire assurant le rôle de serveur DHCP et TFTP, serveur de polices,
- éventuellement, des serveurs applicatifs accessibles par exemple par XDMCP, NX, RDP ou ICA.

Le TP propose quatre manips :

- boot d'un CL avec la distribution de base,
- configuration d'un CL en terminal X,
- configuration d'un CL autonome (kiosque internet),
- configuration d'un CL en client NX.

Dans le cadre de ce TP, un serveur unique sous Fedora 9 (`192.168.1.1`) joue le rôle de serveur d'infrastructure et serveur applicatif (chaque binôme y possède un login, `tpxx`). Chaque binôme dispose :

- d'un PC étiqueté `PCxx` qui sera son CL,
- d'un ou deux portables perso pour accéder en ssh au serveur `192.168.1.1`

Le serveur DHCP est configuré de façon à ce que chaque CL `PCxx` aille chercher ses fichiers dans le répertoire `/tftpboot/tpxx`. Pour plus de détail, voir le fichier de configuration `/etc/dhcpd.conf` du serveur DHCP sur `192.168.1.1`.

Le but des manips est que chaque binôme xx fabrique une arborescence `/tftpboot/tpxx/` comprenant à chaque fois l'ensemble des fichiers nécessaires pour faire démarrer le PC dans la configuration souhaitée.

Tous les fichiers utiles à ce TP se trouvent sur `192.168.1.1` dans le répertoire `/data/thinstation`. Les compilations seront effectuées dans le répertoire `/home/tpxx`, et les fichiers fabriqués seront déposés dans `/tftpboot/tpxx`.

2 Manip #1 : boot de la distribution de base

Il s'agit de fabriquer un noyau `vmlinuz` et un disque virtuel `initrd` pour booter le CL.

[Q1] ⇒ **Sur le serveur `192.168.1.1`, décompressez l'archive de thinstation `/data/Thinstation-2.2.2.tar.gz` dans votre home directory (vous initialiserez une variable d'environnement `DISTDIR` pour plus de commodités).**

```
$ tar xf /data/thinstation/Thinstation-2.2.2.tar.gz -C ~
$ export DISTDIR=~/Thinstation-2.2.2
$ cp /data/thinstation/tg3.ko $DISTDIR/kernel/modules-2.6.16.5/kernel/drivers/net
```

`$DISTDIR/build.conf` est le fichier de définition de la construction. Il est lu par la commande `build`. Y sont énumérés les composants (hardware, applications, fonctionnalités) à charger dans le disque virtuel `initrd` du CL.
La dernière ligne d'instructions remplace le driver réseau fourni par la distribution par une version

plus récente. Cette opération est nécessaire pour prendre en compte des machines munies de chipsets récents.

[Q2] ⇒ **Editez le fichier `$DISTDIR/build.conf` et modifiez-le conformément aux instructions ci-dessous.**

On sélectionne d'abord les modules du noyau à inclure, suivant la syntaxe `module nom-du-module` (pour obtenir une image minimale, on commente tous les modules non cités) :

- driver vidéo : `agpgart` et `intel-agp` pour le chipset intel i810
- driver réseau : `tg3` et `e1000` doivent suffire pour les machines de la salle
- driver son : à voir ; `snd-intel8x0` marche pour certaines machines
- `usb-hid` et `usb-storage` pour les périphériques USB
- support de système de fichiers : `vfat` et `supermount` sont nécessaires pour utiliser les clés USB usuelles ; on peut en rajouter d'autres si on souhaite utiliser des clés formatées avec un autre système de fichiers

(lors de la construction, ces modules seront pris dans le répertoire `$DISTDIR/kernel` de la distribution, voir schéma ci-dessous).

Ensuite on liste les packages thinstation qui seront inclus dans `initrd`, suivant la syntaxe `package nom-du-package` (pour obtenir une image minimale, on commente tous les packages non cités) :

- serveur de son : `sound-esd`
- serveur X : `xorg6-i810` et `xorg6-vesa` suffisent pour les machines de la salle
- package de clavier : choisir le ou les bons claviers
- un client Xterm `rxvt` ou `xterm`
- un gestionnaire de fenêtres `icewm`
- `xtdesk`, pour afficher des icônes sur l'espace de travail
- `samba-server` pour exporter les clés USB

(lors de la construction, ces packages seront pris dans le répertoire `$DISTDIR/packages` de la distribution, voir schéma ci-dessous).

Enfin on initialise quelques paramètres, suivant la syntaxe `param nom-du-parametre valeur` (on modifie les paramètres ci-dessous sans toucher aux autres) :

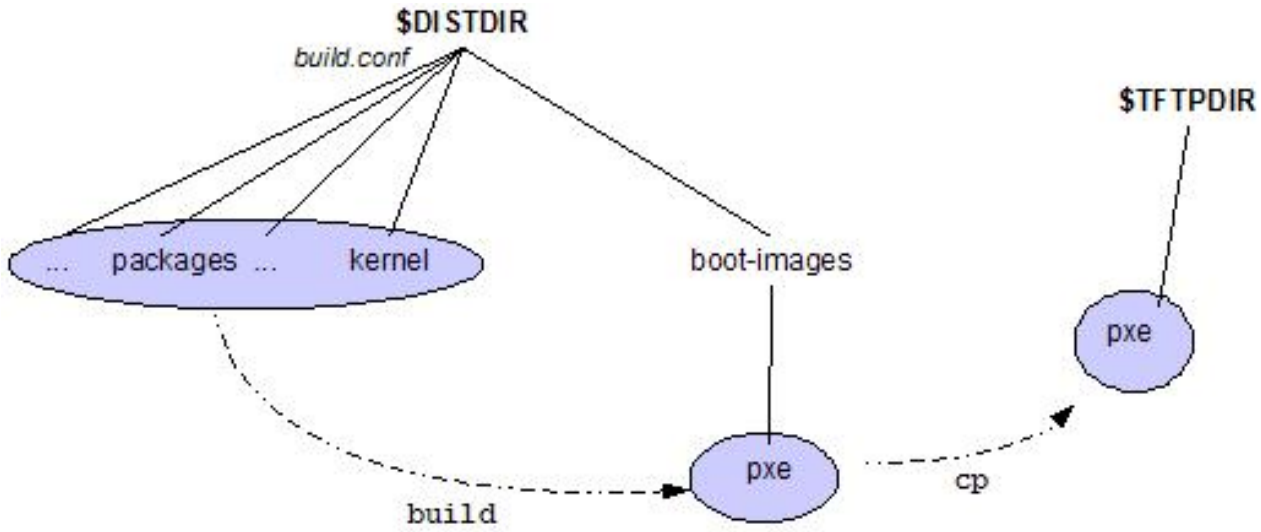
- `param bootlogo false` permet de mieux voir les messages de démarrage
- `param basename TS` : préfix de tous les fichiers de config, qui seront nommés `TS.quelque-chose`
- `param basepath ./tpxx/config`, le chemin d'accès aux fichiers de conf (pour la suite).

Note : les packages listés ci-dessus sont tous disponibles dans la distribution de base, c'est pourquoi une simple ligne `package nom_du_package` suffit pour les inclure dans l'`initrd`. On verra par la suite comment on aurait fait sinon.

On peut maintenant lancer la construction des fichiers de boot par :

```
$ (cd $DISTDIR ; ./build)
```

[Q3] ⇒ **Lancez la construction**



Cela génère des fichiers dans `$DISTDIR/boot-images`, qu'il faut recopier pour les mettre à disposition du serveur TFTP (dans un répertoire `pxe`, pour être conforme à ce qui est spécifié dans la config DHCP) :

[Q4] ⇒ **Créez les répertoires dans l'arborescence tftp et copiez les fichiers créés (initialisez une variable `TFTPDIR` pour faciliter la suite)**

```
$ export TFTPDIR=/tftpboot/$USER
$ mkdir -p $TFTPDIR/{pxe,config,pkg}     # config et pkg, c'est pour la suite !
$ cp -av $DISTDIR/boot-images/pxe/{vmlinuz,initrd,pxelinux.{0,cfg}} $TFTPDIR/pxe
```

[Q5] ⇒ **Démarrez le CL**

Si tout se passe bien, il devrait démarrer (ignorez les messages d'erreur dûs à l'absence du fichier principal de configuration). Vous devriez donc vous retrouver devant un gestionnaire de fenêtre (`icewm`). Vous pouvez déjà lancer un `xterm` et examiner le contenu du CL.

Que s'est-il passé ?
- le CL a chargé par TFTP `pxelinux.0`
- pxelinux a chargé un noyau `vmlinuz` et un disque virtuel `initrd`
- le CL a démarré sur ce noyau/disque virtuel, et a essayé de récupérer des fichiers de configuration : ici, il n'y en avait pas.

Conclusion - Cette manip n'était qu'un petit échauffement : à ce stade, on ne peut en effet pas faire grand-chose. Pour aller plus loin, il va falloir définir des jeux de configuration pour les CL.

2.1 Prise en compte des clés USB

Les clés USB sont montées par le CL, grâce au module `supermount`, dans le répertoire `/mnt/usbdevice`. Ensuite un serveur samba permet au CL d'exporter le contenu de `/mnt/usbdevice` pour une utilisation par un serveur Unix ou Windows.

Les auteurs du TP n'ont pas bien compris la façon dont les clés sont montées dans la version 2.2.2 de Thinstation. Plutôt que de chercher à comprendre, ils ont décidé lâchement de remplacer un fichier de config de la version 2.2.2 par son homologue de la version 2.2.1 (qui a été au préalable copié dans le répertoire `/data/thinstation`) :

```
$ cp /data/thinstation/usb.sh $DISTDIR/packages/base/etc/udev/scripts/usb.sh
```

On va également modifier le comportement du serveur samba, afin que les clés soient exportées en read-write (par défaut, elles ne sont exportées qu'en lecture seule). Pour cela, on remplace le fichier `smb.conf.tpl` par défaut par un `smb.conf.tpl` minimal de notre crû :

```
$ cp /data/thinstation/smb.conf.tpl $DISTDIR/packages/samba-base/lib
```

Note : cette méthode n'est pas secure, car tout le réseau à accès en lecture/écriture à la clé ! On peut faire mieux, mais c'est plus compliqué.

[Q6] ⇒ **Effectuer les modifications ci-dessus, relancer une construction par `build`, recopier le `initrd` ainsi construit à sa place dans `$TFTPDIR`, et redémarrer le terminal.**

```
$ (cd $DISTDIR ; ./build)
$ cp $DISTDIR/boot-images/pxe/initrd $TFTPDIR/pxe
```

On pourra tester avec une clé USB qu'elle est bien montée automatiquement dans `/mnt/usbdevice`.

3 Manip #2 : configuration d'un terminal X

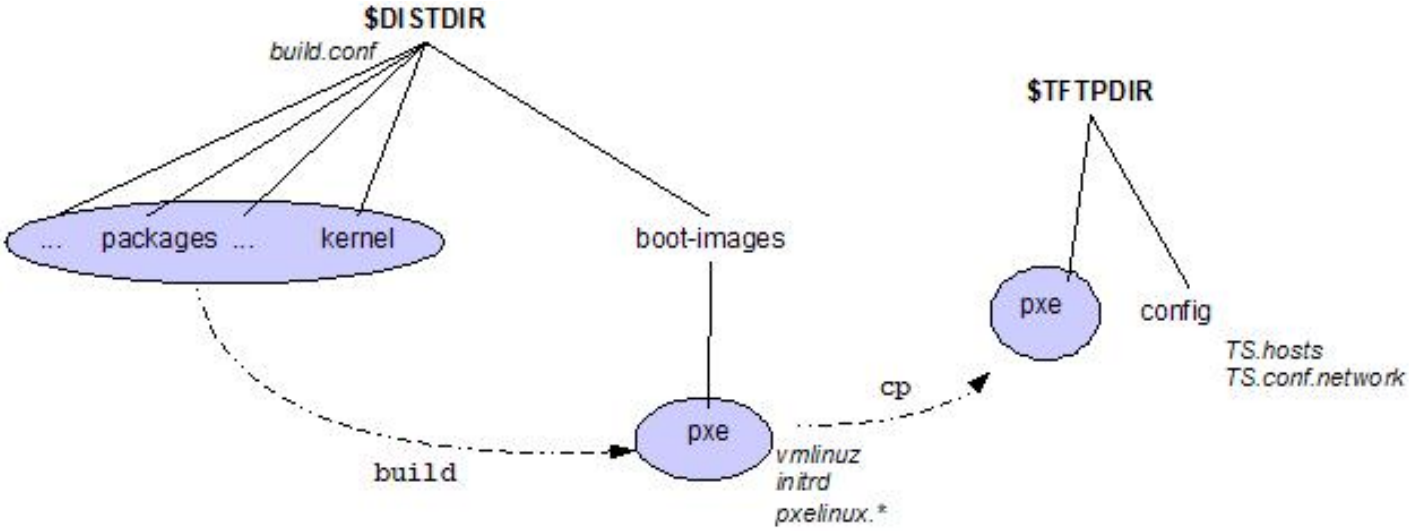
Les fichiers de configuration vont permettre de spécifier des paramétres pour chaque client. Suivant le fichier dans lequel il est défini, un paramétre sera commun à tous les clients, spécifique à un client donné, ou spécifique à un groupe de clients. Les paramètres peuvent être spécifiés dans n'importe quel fichier, et on peut créer autant de fichiers de conf que l'on veut.

La distribution fournit un fichier `thinstation.conf.sample`, contenant l'ensemble des paramètres utilisables.

On distingue deux types de fichiers de configuration :

- `$DISTDIR/thinstation.conf.buildtime`, qui est inclus en dur dans le fichier `initrd` à sa fabrication,
- les autres, `TS.hosts` et `TS.conf.network`, mis à disposition sur le serveur TFTP (dans le répertoire spécifié par le paramètre `basepath`, initialisé lors de la manip #1 à `$TFTPDIR/config`), et qui sont récupérés par les CL au boot.

On peut créer d'autres fichiers de ce dernier type, `TS.conf-xxxx` pour paramètrer un CL particulier et `TS.conf.group-xxxx` pour paramètrer un ensemble de CL.



Plus précisément :

- `TS.hosts`, s'il est présent, associe à chaque adresse MAC un nom de machine (pas forcément celui du DNS) et un certain nombre de groupes auxquels appartient ce CL. Chaque CL est référencé par une ligne de la forme
  `tp01 001ec9360931 hres x11 nx`
  qui signifie que la machine `tp01` possède l'adresse MAC `001ec9360931` et appartient aux groupes `hres`, `x11` et `nx`.
- `TS.conf.network`, s'il existe, précise des paramètres communs à tous les CL
- `TS.conf-tp01` : paramètres pour la machine `tp01` (où `tp01` est un nom de machine défini dans `TS.hosts`)
- `TS.conf-001ec9360931` ou `TS.conf-192.168.1.11` : paramètres pour une machine d'adresse MAC ou d'adresse IP donnée
- `TS.conf.group-hres` paramètres pour le groupe de machines `hres` (où `hres` est un groupe de CL défini dans `TS.hosts`)

Dans ce qui suit, on va produire `TS.hosts`, `TS.conf.network`, et trois fichiers de groupes : `TS.conf.group-lres`, `TS.conf.group-hres` et `TS.conf.group-X11`.

1. Le fichier `TS.hosts`
   [Q7] ⇒ **Récupérez l'adresse MAC de votre CL et créez un fichier** `$TFTPDIR/config/TS.hosts`**.**

```
$ grep $USER /etc/dhcpd.conf
$ echo "$USER adresse-mac-aabbccddeeff lres X11" > $TFTPDIR/config/TS.hosts
```

2. Configuration commune à tous les CL
   [Q8] ⇒ **Créez un fichier** `$TFTPDIR/config/TS.conf.network` **contenant les paramètres énumérés ci-dessous**

```
KEYBOARD_MAP=fr
USB_ENABLED=On
AUTOSTART=On
PKG_PREFIX=../pkg
SCREEN_X_FONT_SERVER=192.168.1.1:7100

SESSION_0_TYPE=icewm
SESSION_0_SCREEN=0

SAMBA_SERVER_ENABLED=On
SAMBA_USB=On
```

3. Configuration de groupes
   On va définir deux groupes `hres` et `lres` pour adapter la résolution aux desiderata des différents utilisateurs. De même, on va définir un groupe `X11` pour paramétrer les CL destinés à se connecter à un serveur XDM.

   [Q9] ⇒ **Créez deux fichiers** `$TFTPDIR/config/TS.conf.group-{hres,lres}` **contenant chacun respectivement une ligne :**

```
SCREEN_RESOLUTION=1280x1024
```

   et

```
SCREEN_RESOLUTION=1024x768
```

   [Q10] ⇒ **Créez un fichier** `$TFTPDIR/config/TS.conf.group-X11` **contenant les lignes suivantes :**

```
SESSION_1_TYPE=x
SESSION_1_SCREEN=1
SESSION_1_X_SERVER=192.168.1.1
SESSION_1_X_OPTIONS="-query"
```

A ce niveau, le CL doit pouvoir ouvrir une session X11 sur le serveur interactif.

[Q11] ⇒ **Redémarrez le CL et testez**

Que s'est-il passé ? En fin de boot, le CL a récupéré les fichiers `TS.hosts`, `TS.conf.network`, en a déduit qu'il fallait ensuite charger `TS.conf.group-lres` et `TS.conf.group.X11`, ce qui l'a conduit à ouvrir une session XDM sur `192.168.1.1`.

[Q12] ⇒ **Remplacez** `lres` **par** `hres` **dans le fichier** `TS.hosts`**, et rebooter le client.**

Un CL peut se connecter simultanément à plusieurs serveurs interactifs.

[Q13] ⇒ **Ajoutez dans** `TS.conf.group-X11` **les lignes suivantes, et rebooter le CL.**

```
SESSION_2_TYPE=x
SESSION_2_SCREEN=2
SESSION_2_X_SERVER=192.168.1.2
SESSION_2_X_OPTIONS="-query"
```

[Q14] ⇒ **Jouez avec les touches** `Ctrl Alt F{1,2,3,4,5}`

Conclusion - L'installation d'une telle architecture prend un certain temps de mise au point. Mais une fois opérationnelle, l'installation d'un nouveau poste de travail se résume à :

- sortir le PC du carton
- retirer le disque dur (optionnel)
- brancher le secteur, le réseau et récupérer l'adresse MAC
- ajouter une entrée au serveur DHCP et une entrée au fichier TS.hosts
- booter

soit 5 mn maximum. Et zéro intervention sur le poste utilisateur par la suite...

3.1 Configuration vue du côté utilisateur

Dans cette section, on va modifier les fichiers de configuration de l'utilisateur pour qu'il puisse utiliser le son et les clés USB depuis le serveur. Pour ceci, on va ajouter dans son fichier `.bashrc` les lignes :

```
TERMINAL=‘echo $DISPLAY|cut -f1 -d:‘
alias cleusb=’nautilus --no-desktop --browser smb://${TERMINAL}:16001’
export ESPEAKER="$TERMINAL:16001"
```

Note : dans la vraie vie, l'administrateur pourrait faire ça sous forme d'un shell script déposé par exemple dans `/usr/local/bin` sur chaque serveur applicatif (mais pour le TP, ce n'est pas possible).

Explications :

- la deuxième ligne définit une commande `cleusb` qui demande au gestionnaire de fichiers `nautilus` de faire un montage SMB sur le terminal afin de monter la clé
- la troisième ligne dit au serveur interactif d'envoyer le son sur le serveur ESD du terminal

Il reste encore à demander aux applications d'utiliser `esd`. Ainsi, pour configurer `mplayer`, on crée un fichier `~/.mplayer/config`, contenant les lignes :

```
[default]
ao=esd
```

[Q15] ⇒ **Ouvrez une nouvelle session X sur le server `192.168.1.1`, et testez le montage des clés USB et le son.**

On pourra par exemple se connecter au site `http://franceinter.com` avec `firefox`, et écouter le direct (`firefox` utilise pour cela le plugin `mplayer`).

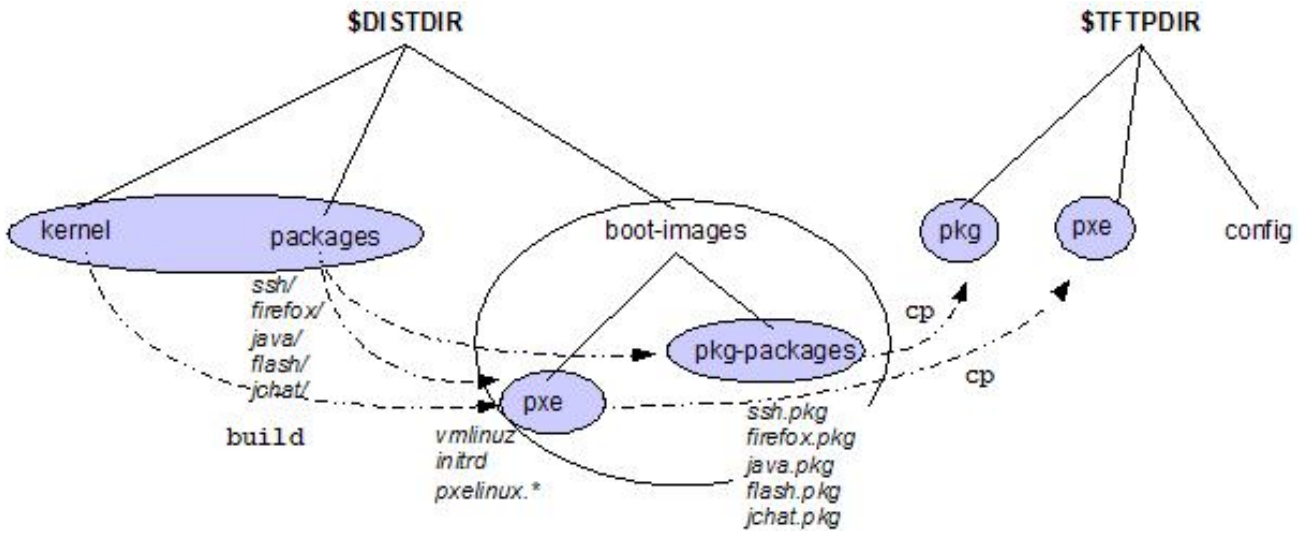
4 Manip #3 : configuration d'un CL autonome (kiosque internet)

On veut mettre en place un accès internet anonyme (web, chat et ssh). Pour cela, nous allons transformer le CL en un *kiosque internet*, et le doter de toutes les applications dont il a besoin pour être autonome : `ssh`, `firefox`, avec plugins flash et java, et `jchat`. A l'occasion (ça ne coûte pas plus cher, et ça peut toujours servir), on va mettre aussi un client RDP.

On a le choix entre :

- ajouter les applis souhaitées à l'`initrd`, comme nous l'avons fait à la manip #1 avec la directive `package`,
- conserver un `initrd` minimal commun à tous les CL, et télécharger les applis dynamiquement, sur certains CL, avec la directive `pkg`.

C'est cette dernière solution que nous allons mettre en oeuvre. Nous allons donc d'abord construire des packages, puis écrire un fichier de configuration qui fait appel à ces packages.



Nous allons donc déclarer cinq nouvelles applications, en ajoutant les lignes suivantes au fichier `$DISTDIR/build.conf` :

```
pkg ssh
pkg java
pkg firefox
pkg flash
pkg jchat
pkg rdesktop
pkg extra-fonts-75dpi
```

Les applis sélectionnées sont de quatre sortes :

- ssh et `rdesktop`, qui sont inclus dans la distribution : il existe des répertoires `$DISTDIR/packages/ssh` et `$DISTDIR/packages/rdesktop` complet. Par conséquent une simple déclaration `pkg ssh` et `pkg rdesktop` dans `$DISTDIR/build.conf` suffit.
- java, firefox, flash, qui sont des applications prévues dans la distribution, mais (pour des raisons de licences ou autre) ne sont pas incluses : il existe pour chacune d'elles, dans `$DISTDIR/packages`, un répertoire incomplet, car les binaires n'y sont pas. Il faut donc indiquer où ceux-ci se trouvent (pour les besoins du TP, on les a téléchargés dans `/data/thinstation`). Pour cela, il suffit d'ajuster les valeurs des paramètres suivants dans le fichier `$DISTDIR/build.conf` :

```
param firefoxurl     file://data/thinstation/firefox-2.x-current.tar.gz
param flashurl       file://data/thinstation/install_flash_player_9_linux.tar.gz
param javaurl        file://data/thinstation/jre-1_5_0_17-linux-i586.bin
param nxurl          file://data/thinstation/nxclient-3.3.0-6.i386.tar.gz
```

  (NX ne sera pas utile tout de suite, mais ça sera fait !)
- `extra-fonts-75dpi` est un package de polices de caractères supplémentaires, utiles pour afficher correctement `firefox`. Il ne fait pas directement partie de la distribution, et devra être rajouté dans le répertoire `$DISTDIR/package`.

```
$ tar xf /data/thinstation/extra-fonts-75dpi.tar.gz -C $DISTDIR
```

- enfin, `jchat`, qui n'a pas été prévu du tout dans la distribution : nous allons créer à la main le package ; c'est l'occasion de regarder comment est constitué un package.

[Q16] ⇒ **Ajoutez les 7 définitions `pkg` ci-dessus dans le fichier `build.conf`, les 4 initialisations `param`, et ajouter le package de polices de caractères**

Pour créer le package jchat, on commence par récupérer une archive de l'exécutable `jchatirc` :

```
$ tar xf /data/thinstation/jchatirc-2.6-6.tar.gz -C ~
```

qui n'est autre qu'une archive java `JChatIRC.jar`.

On va créer alors une arborescence `$DISTDIR/packages/jchat`,

```
$ mkdir -p $DISTDIR/packages/jchat/{lib/menu,usr/icons,bin,etc/{cmd,init.d}}
```

ayant la structure suivante :

```
jchat
jchat/dependencies
jchat/lib
jchat/lib/JChatIRC.jar
jchat/lib/menu
jchat/lib/menu/jchat
jchat/usr
jchat/usr/icons
jchat/usr/icons/jchat_32x32.xpm
jchat/bin
jchat/bin/jchat
jchat/etc
jchat/etc/cmd
jchat/etc/cmd/jchat.fullscreen
jchat/etc/init.d
jchat/etc/init.d/jchat
```

Dans cette arborescence,

- `dependencies` est un fichier texte décrivant les dépendances du package. Dans notre cas, il contiendra deux lignes contenant chacune un unique mot, à savoir respectivement `base` et `java`.
- Les fichiers `jchat_32x32.xpm` et `JChatIRC.jar` sont des copies des fichiers fournis dans l'archive de `jchat`
- `bin/jchat` est un fichier texte exécutable, contenant la ligne
  ```
  java -jar /lib/JChatIRC.jar
  ```
- `jchat.fullscreen` contient la ligne
  ```
  CMD_FULLSCREEN="java -jar /lib/JChatIRC.jar"
  ```
- `menu/jchat` contient la ligne
  ```
  package="jchat"; needs="x11"; title="jchat"; command="java -jar /lib/JChatIRC.jar"
  ```
- `etc/init.d/jchat` est un lien symbolique vers `/etc/thinstation.packages`.

[Q17] ⇒ **Créez le package `jchat`**

On va ensuite modifier légèrement le package de firefox pour que firefox s'ouvre par défaut sur une page adaptée aux mathématiciens. Pour cela, il suffit de modifier dans le fichier `$DISTDIR/packages/firefox/etc/rc5.d/S10firefox.init` la variable `FF_HOMEPAGE`, pour y mettre `http://math.cnrs.fr`.

[Q18] ⇒ **Effectuez le paramétrage de firefox et lancez la construction**

Il faudra plusieurs fois répondre `yes` pour accepter les licences.

Cette construction construit, à partir de `$DISTDIR/package` des packages (fichiers d'extension `.pkg`) dans `$DISTDIR/boot-images/pkg-packages`, qu'il restera à recopier pour mettre à disposition du serveur TFTP (dans le répertoire spécifié par le paramètre `PKG_PREFIX`, initialisé lors de la manip #2 à `$TFTPDIR/pkg`) :

```
$ cp -v $DISTDIR/boot-images/pkg-packages/* $TFTPDIR/pkg/
```

On crée ensuite un nouveau fichier de configuration de groupe `$TFTPDIR/config/TS.conf.group-kiosk` contenant la ligne :

```
PKG_PACKAGES="rdesktop firefox flash java ssh jchat extra-fonts-75dpi"
```

et on modifie le `$TFTPDIR/config/TS.hosts` en remplaçant le groupe `X11` par `kiosk` :

```
tpxx adresse-mac-de-tpxx hres kiosk
```

[Q19] ⇒ **Mettez à jour `$TFTPDIR/pkg` et `TS.hosts`, et créez `$TFTPDIR/config/TS.conf.group-kiosk`**

[Q20] ⇒ **Rebootez le CL et testez.**

Vous pouvez lancer une session `rdesktop` en cliquant sur l'icône windows, et en rentrant comme serveur `172.19.45.12` (vous obtiendrez une mire de connexion, mais vous ne pourrez pas lancer de

session sur le serveur windows, car on ne vous a pas créé de compte sur cette machine).

Conclusion - Vos invités de passage veulent tous sans exception pouvoir consulter leur webmail ou lancer leur `pine` à distance. Ils (ceux qui n'ont pas encore de portable) vous demandent donc naturellement un compte sur les machines de votre labo. Si vous avez ras-le-bol d'assurer cette tâche ingrate, le kiosque internet est une alternative...

5 Manip #4 : configuration d'un client NX

Préambule - Les auteurs du TP n'étant pas des pros de NX, il est possible que vous assistiez au cours de cette manip à des phénomènes inexpliqués (mais ils espèrent pouvoir compter sur la collaboration des gourous NX présents dans l'assistance).

NX est une solution de type Terminal Serveur permettant de lancer n'importe quelle application X sur n'importe quel système d'exploitation à travers le réseau.

Le but de la manip est de doter le CL d'un client NX configuré pour ouvrir une session NX sur le serveur interactif.

Les binaires NX ne sont pas inclus dans Thinstation, mais NX est prévu (cas similaire à firefox, flash et java de la manip #3). On va donc :

- inclure les binaires au package (ajout d'une ligne `pkg nx` dans `build.conf`, puis premier `build`),
- personnaliser les composants du package,
- construire le package (via un second `build`).

La personnalisation va consister à inclure une clé publique et un fichier de configuration de session.

En effet, le client NX établit au préalable une connexion `ssh` authentifiée par un couple clé privée/clée publique. La clé que nous allons utiliser a été créée sur le serveur interactif en lançant la commande `/usr/libexec/nx/nxkeygen`.

Par ailleurs, chaque connexion NX est paramétrée par un fichier `nonDeLaSession.nxs`. Un tel fichier, nommé `NX.nxs` et adapté à notre serveur, a été fabriqué au préalable en lançant d'abord un client NX non configuré sur un client léger, et en utilisant l'outil de configuration graphique de nxclient.

Les deux fichiers `server.id_dsa.key` et `NX.nxs` sont disponibles dans le répertoire de données du TP. Il suffit donc de les installer au bon endroit, c'est-à-dire respectivement dans les répertoires `usr/NX/share/keys` et `etc/nx` du package `nx` de la distribution.

Par défaut, le CL affiche trois icônes permettant à l'utilisateur de modifier les réglages. Comme on souhaite avoir une configuration ne risquant pas d'induire un utilisateur peu averti en erreur, on va supprimer cette fonctionnalité, en effaçant simplement le fichier `lib/menu/nx` du package `nx` de la distribution.

Cela fait, on peut lancer la construction finale, et recopier le package `nx.pkg` obtenu dans l'arborescence tftp.

[Q21] ⇒ **Effectuez la construction décrite ci-dessus.**

Maintenant, on modifie la configuration pour que le CL utilise ce package. On crée un fichier `TS.conf.group-nx` contenant la ligne `PKG_PACKAGES1="nx"`, et on ajoute le groupe nx dans le fichier `TS.hosts`.

[Q22] ⇒ **Ajustez la config, redémarrez le CL et testez une connexion NX sur le serveur.**

Conclusion - Basé sur la compression X et un système de caches, le protocole NX rend la session aussi réactive que possible. Essayez par exemple de lancer la commande `xmaple` sous NX et comparez le temps de réponse avec celui observé lors d'une connexion X.